# Project Assignment for the [ASP.NET MVC Course @ SoftUni](https://softuni.bg/trainings/1537/csharp-mvc-frameworks-asp-net-march-2017)

## General Requirements

Your Web application should use the following technologies, frameworks and development techniques:

* The application must be implemented using **ASP.NET MVC** framework.
  + The application must have at least 8 web pages (views) **[done]**
  + The application must have at least 4 entity models **[done]**
  + The application must have at least 4 controllers **[done]**
* Use **Visual Studio 2015 or 2017**.
  + Use **Razor** template engine for generating the UI.
    - Rendering with ASP.NET Web Forms is **not** allowed.
    - Use **sections** and **partial views**. **[done]**
    - Use **editor** and **display templates**. **[done]**
  + You could also make the front-end using JavaScript and consuming a rest service from Web API
* Use **Microsoft SQL Server** as database back-end. **[done]**
* Use **Entity Framework 6** to access your database. **[done]**
* Use **MVC Areas** to separate different parts of your application (e.g. area for administration). **[done]**
* Adapt the default **ASP.NET MVC site template** or get another free theme.
  + Use responsive design based on **Twitter Bootstrap/Google Material design**. **[done] (Kendo UI)**
* Use the standard **ASP.NET Identity System** for managing **users** and **roles**. **[done]**
  + Your registered users should have at least one of these roles: **user** and **administrator**. **[done]**
* Use **AJAX** request to asynchronously load and display datasomewhere in your application. **[done]**
* Write **unit tests** for your logic, controllers, actions, helpers, etc.
* Implement **error handling** and **data validation** to avoid crashes when invalid data is entered (both **client-side** and **server-side**).
* Handle correctly the special HTML characters and tags like **<br />** and **<script> (escape special characters).**
* **Use Ninject (or any other dependency injection container).**  **[done]**
* **Use AutoМapper.** **[done]**
* **Prevent from security vulnerabilities like SQL Injection, XSS, CSRF, parameter tampering, etc.**

## Additional Requirements

* Follow the best practices for OO design and **high-quality code** for the Web application:
  + Use data encapsulation.
  + Use exception handling properly.
  + Use inheritance, abstraction and polymorphism properly.
  + Follow the principles of strong cohesion and loose coupling.
  + Correctly format and structure your code, name your identifiers and make the code readable.
* Well looking user interface (UI).
* Good usability (easy to use UI).
* Supporting of all modern Web browsers.
* Use caching where appropriate.
* Use a **source control system** by choice, e.g. GitHub, BitBucket.
  + Submit a link to your public source code repository.

## Public Project Defense

Each student will have to deliver a **public defense** of its work in front of the other students, trainers and assistants. Students will have **only 15 minutes** for the following:

* **Demonstrate** how the application works (very shortly).
* Show the **source code** and explain how it works.
* Optionally you might prepare a **presentation** (3-4 slides).

Please be **strict in timing**! On the 15th minute you **will be interrupted**! It is good idea to leave **the last 2-3 minutes for questions** from the other students, trainers and assistants.

Be **well prepared** for presenting maximum of your work for minimum time. Bring your own laptop. Test it preliminary with the multimedia projector. Open the project assets beforehand to save time.

## Bonuses

* Use **SignalR** communication somewhere in your application.
* Host the application in a **cloud environment**, e.g. in **AppHarbor** or **Azure**.
* Use a **file storage cloud API**, e.g. **Dropbox**, **Google Drive** or other for storing the files.
* Use of features of HTML 5 like Geolocation, Local Storage, SVG, Canvas, etc.
* Anything that is not described in the assignment is a bonus if it has some practical use.

## Assessment Criteria

* **Functionality** – **0…20**
* **Implementing controllers correctly** (controllers should do only their work) **– 0...5**
* **Implementing views correctly** (using display and editor templates) **– 0…5**
* **Unit tests** (unit test for some of the controllers using mocking) **– 0…10**
* **Security** (prevent SQL injection, XSS, CSRF, parameter tampering, etc.) **– 0…5**
* **Data validation** (validation in the models and input models) **– 0…10**
* **Using auto mapper and inversion of control – 0…5**
* **Using areas with multiple layouts – 0…10**
* **Code quality** (well-structured code, following the MVC pattern, following SOLID principles, etc.) – **0…10**
* **Bonus** (bonus points are given for exceptional project) – **0…25**